

Lawrence Livermore Laboratory

LIEPROC:

A MACSYMA Program for Finding Adiabatic Invariants of Simple Hamiltonian Systems

via the Lie Transform

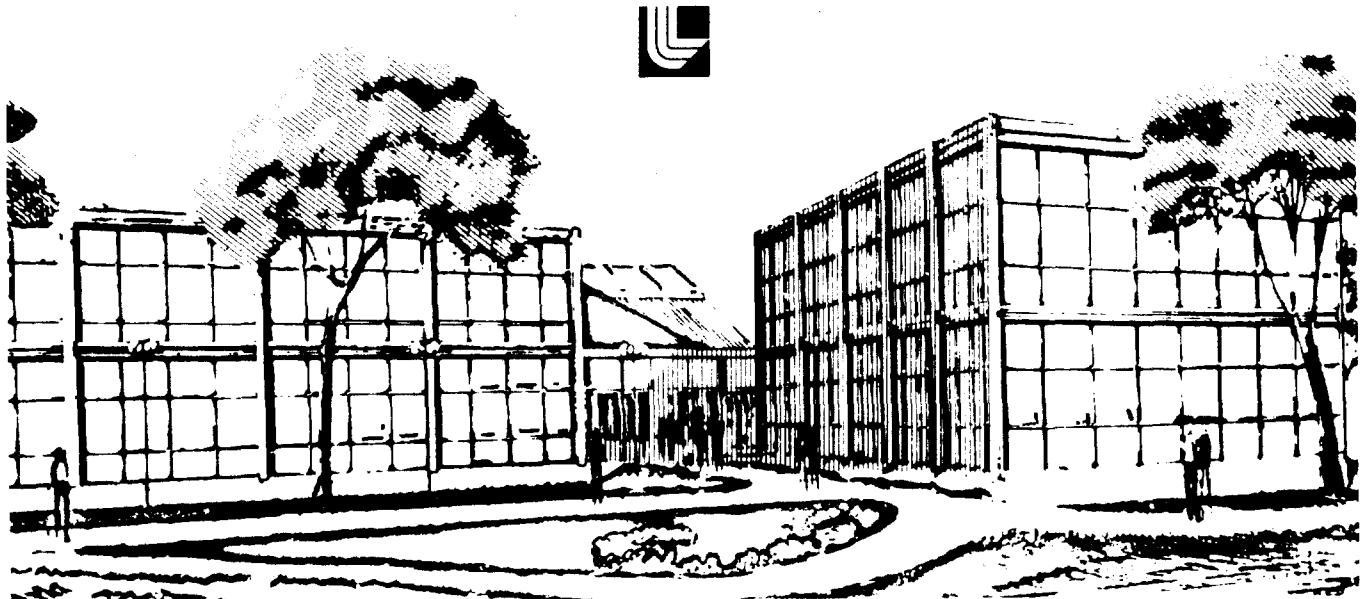
Bruce Char and Brendan McNamara

November 22, 1978

CIRCULATION COPY
SUBJECT TO RECALL
IN TWO WEEKS

This paper was prepared for the Second MACSYMA User's Conference, June 20-22, 1979,
Washington, D.C.

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.



DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

LIEPROC:
A MACSYMA[†] Program for Finding Adiabatic Invariants
of Simple Hamiltonian Systems via the Lie Transform

Bruce Char

University of California at Berkeley*
Computer Science Division
EECS Department
Berkeley, California 94720

and

Brendan McNamara

Lawrence Livermore Laboratory
University of California, P. O. Box 808
Livermore, California 94550

ABSTRACT

The usage and performance of a program in a symbolic manipulation language that computes adiabatic invariants of certain Hamiltonian systems via the Lie transform is discussed.

[†]MACSYMA is a product of the Mathlab group of the Laboratory for Computer Science, Massachusetts Institute of Technology, supported in part by the Department of Energy under contract number E (11-1)-3070 and by the National Aeronautics and Space Administration under Grant NSG 1323.

*Work performed while summer visitor at Lawrence Livermore Laboratory.
Work performed under the auspices of the U.S. Dept. of Energy by the Lawrence Livermore Laboratory under contract number W-7405-ENG-48.

TABLE OF CONTENTS

	Page
Introduction	1
The Lie Algorithms	2
Description of Procedure	6
Performance	6
Acknowledgements	9
References	10
Appendix A Usage of Procedure Lieproc	12
Appendix B Sample Output of Lieproc	17
Appendix C Listing of MACSYMA Procedures for Lie Transform	25

Introduction

The perturbation theory of Hamiltonian systems made a major step forward with the introduction of the Lie transform technique by Deprit [1] and described by Nayfeh [2]. A single scalar generating function is calculated to give a canonical transformation of the Hamiltonian, the coordinates, and any function of the coordinates from old to new or new to old coordinates. When the perturbations are periodic along the unperturbed orbits, the Hamiltonian can be averaged, an adiabatic invariant may exist, and the averaged equations of motion may be considerably simplified. The invariants of the motion have played a key role in many plasma devices and continue to be of interest for the containment of plasma, the structure of magnetic surfaces in toroidal devices, and the propagation of large amplitude waves for plasma heating. Several Lie transforms appropriate to such problems are given by McNamara [3,1], and a review of the theory of charged particle motion is given in ref. 11.

The algebra of the methods depends upon the three operators - Poisson bracket, average along an orbit, and integral along an orbit. A convenient form of the algorithm is described by Nayfeh [2], and we have expressed it in the notations used by McNamara [3,4]. The algorithms and operators are not defined in the MACSYMA language [6] although most of the algebraic manipulations of simplification, differentiation, subscripting , and trigonometry are available. This paper describes a MACSYMA program to implement the Lie transform algorithms, in the simplest cases of fixed frequency perturbations, and demonstrates that the pattern matching and user property definition capabilities are sufficient to carry through these very different procedures.

The Lie Algorithms

Given a system of differential equations

$$x = f(x, \epsilon) = \sum_{i=0}^{\infty} \frac{\epsilon^i}{i!} f_i(x)$$

the Lie Transform method produces approximations to a solution to a derived system of equations,

$$y = y(x, \epsilon) = \sum_{i=0}^{\infty} \frac{\epsilon^i}{i!} g_i(x)$$

where the map from x to y is a near identity transformation

$$y = x + \sum_{i=0}^{\infty} \frac{\epsilon^i}{i!} x_i(x, \epsilon) \text{ with inverse map}$$

$$x = y + \sum_{i=0}^{\infty} \frac{\epsilon^i}{i!} y_i(y, \epsilon).$$

The basis of the Lie Transform is the introduction of another set of equations

$$\frac{dx}{d\epsilon} = w(x, \epsilon) = \sum_{i=0}^{\infty} \frac{\epsilon^i}{i!} w_i(x) \text{ where } x(\epsilon=0) = y$$

which are easier to solve for small ϵ than the original system is to solve for large t . As developed by Deprit [1] and presented in Nayfeh [2], this method can be used in an algorithm for computing g_i , x_i , and w_i given f , and sufficient specifications on g or w .

In particular, the Lie Transform can be applied to the problem of finding invariants of Hamiltonian systems:

Given a time independent Hamiltonian H

$$H(p, q, \varepsilon) = \sum_{i=0}^{\infty} \frac{\varepsilon^i}{i!} H_i(p, q)$$

where p and q each are
m-dimensional vectors in a
(p, q) coordinate system X

pick the generating function W

$$W = \begin{bmatrix} S_p \\ S_q \\ 0 \end{bmatrix}$$

for a new coordinate system $Y = (P, Q)$ with Hamiltonian

$$K(P, Q, \varepsilon) = \sum_{i=0}^{\infty} \frac{\varepsilon^i}{i!} K_i(P, Q) \quad (P, Q \text{ also m-dimensional}).$$

so that the K_i have no terms involving Q_1 . In the previous notation, we have

$$x = \begin{bmatrix} p \\ q \\ t \end{bmatrix} \quad y = \begin{bmatrix} P \\ Q \\ T \end{bmatrix}$$

$$f = \begin{bmatrix} -\frac{\partial H}{\partial q} \\ -\frac{\partial H}{\partial p} \\ 0 \end{bmatrix} \quad g = \begin{bmatrix} -\frac{\partial K}{\partial Q} \\ -\frac{\partial K}{\partial P} \\ 0 \end{bmatrix}$$

Thus, given H , the generating functions S and K , can be computed as well as the coordinate maps

$$P(p, q, \varepsilon) = p + \sum_{i=0}^{\infty} p^i(p, q) \frac{\varepsilon^i}{i!}$$

$$Q(p, q, \varepsilon) = q + \sum_{i=0}^{\infty} q^i(p, q) \frac{\varepsilon^i}{i!}$$

and their inverses $p(P, Q, \varepsilon)$ and $q(P, Q, \varepsilon)$.

As in Nayfeh [2], pp. 201-216, and McNamara [3], this particular case leads to the following:

$$K_0(P, Q) = H_0(P, Q)$$

$$T_n(x, y) = H_n(x, y) + \sum_{j=1}^{n-1} \binom{n-1}{j-1} [H_{n-j}(x, y), S_j(x, y)] \\ + \binom{n-1}{j} K_{j, n-j}(x, y)$$

$$S_n = ii(T_r) \quad K_n = av(T_r) \quad K_{j, i} = [K_i, S_j] - \sum_{m=1}^{j-1} \binom{j-1}{m-1} [K_{j-m, i}, S_i]$$

where $[f, g]$ is the Poisson bracket operator

$$[f, g](P, Q) = \sum_{i=1}^n \frac{\partial f}{\partial Q_i} \frac{\partial g}{\partial P_i} - \frac{\partial f}{\partial P_i} \frac{\partial g}{\partial Q_i}$$

and ii and av are operators designed to remove q_1 terms from the K_j :

$$ii(f) = \int_0^y f(p, q) dq_1 \quad \text{where } y \text{ is the period of } f \text{ in } q_1$$

$$av(f) = \int (f - ii(f)) dq_1$$

The inverse transform from (P, Q) to (p, q) is defined by

$$q(P, Q) = \sum_{i=0}^{\infty} q^i(P, Q) \frac{\epsilon^i}{i!}$$

$$p(P, Q) = \sum_{i=0}^{\infty} p^i(P, Q) \frac{\epsilon^i}{i!}$$

where:

$$q^0 = Q \quad q^n = \frac{\partial S_n}{\partial p} + \sum_{j=1}^{n-1} \binom{n-1}{j} q_j, n-j$$

$$q_{j,i} = [q^{(i)}, s_j] = \sum_{m=1}^{j-1} \binom{j-1}{m-1} [q_{j-m,i}, s_m]$$

and

$$p^0 = P \quad p^{(n)} = \frac{\partial S_n}{\partial Q} + \sum_{j=1}^{n-1} \binom{n-1}{j} p_{j,n-j}$$

where $p_{j,i}$ is of a form similar to $q_{j,i}$.

The transform from (p,q) to (P,Q) is given by

$$p^n = -p^n + \sum_{j=1}^{n-1} \binom{n}{j} p_{j,n-j}$$

$$Q^n = -q^n + \sum_{j=1}^{n-1} \binom{n}{j} q_{j,n-j}$$

Description of Procedure

A series of MACSYMA [4] routines were written to compute and print order by order the transformed Hamiltonian K, the generating function S, and the coordinate maps. The programs were designed to be used in the following way:

- a) Upon starting up MACSYMA, load the file containing the LT routines.
- b) Enter the Hamiltonian H (as a finite series in powers of epsilon) and two lists describing the name and dimensionality of the origin (X) and target spaces (Y) of the transform.
- c) Call the main procedure lieproc specifying among other parameters, the order n describing the number of orders to which the transform is to be computed. Once called, the routine calculates and prints its results non-interactively until it completes order n or encounters an error condition. Since MACSYMA runs interactively, extensive editing, interruptions, expression, manipulation, etc. are possible before, during, and after the computation. See Appendix A for an example of a sample MACSYMA session invoking the LT procedures. Appendix B contains the program listings for the routines involved.

Performance

Since MACSYMA is currently supported only on the Macsyma Consortium system (available through the DCA Arpanet at the Massachusetts Institute of Technology's Laboratory for Computer Science, in Cambridge, Massachusetts), discussion of the actual performance of the routine will be limited to the Consortium's computer, a Decsystem-10 (KL Model). This machine has a basic instruction cycle of approximately 700 nanoseconds, with approximately 128,000 36-bit words of primary storage available to MACSYMA users for their programs and data. Since the MACSYMA system is written in LISP, part of the execution time is spent doing LISP "garbage collection", as opposed to execution of MACSYMA programs.

Table I contains timing and space requirements for the computation to fourth order for the simple Hamiltonian

$$H(p,q) = p_1 + (p_1 + p_2) [1 - \cos(2\alpha[q_1 + q_2]) \sin(q_1 + q_2)]$$

TABLE I

Expression	Order	Time to compute (in seconds)	Size of expression+
S	1	10	649
	2	190	2836
	3	417	6808
	4	2910	16135
q ⁱ	1	6.8	2279
	2	276	7254
	3	*	*
	4	*	*

+ Measured in number of characters (print length in expression when saved as a MACSYMA SAVE file. Intended as rough measure of size.

* Computation unable to proceed at this point -- out of memory.

The present structure of the program is sufficient to allow implementation of LIEPROC for the case of Hamiltonians with resonant denominators (see McNamara [4]). However, the program currently suffers from two limitations: poor utilization of functional identities (simplification of functional expressions), and inherent space limitations on the size of expressions in the MACSYMA system. We discuss each of these problems and the prospects for their solution individually.

LIEPROC evaluates where it could save time by, instead of simplifying functional expressions, using rules such as the Jacobi Poisson bracket identities and the ii and av operator algebra identities (see McNamara and Whiteman [9]). The problem is that MACSYMA is geared toward the manipulation of general mathematical expressions (e.g. $2*a + a$ simplifies into $3*a$); it is harder to have MACSYMA simplify functional expressions - e.g. in LISP/MACSYMA notation $(\lambda(x)x) + (\lambda(x)2*x)$.

simplified into $(\lambda(x) 3*x)$ or $3*(\lambda(x) x)$. Since functional expressions are the primary result of the Lie Transform (S, K, P, Q, p and q are all functions in the current implementation), future improvements in speed are to be expected when functional simplification rules are included in the code. Alternatively, one could strip away functional notation from the algorithm wherever possible, reducing the functional simplification process to the easier case of ordinary algebraic simplification. This would put the results of the program in a technically incorrect form (an "abuse of notation" convention would be imposed upon the user). Adopting this scheme would also force the evaluation of all functions before simplification, which a functional simplification scheme would avoid.

The other, more serious problem encountered during LIEPROC calculations is the space (memory) requirements of the computation. Currently, the memory space available for expressions (approximately 70,000 words) is exceeded for calculations involving all but the most trivial Hamiltonians beyond third order. To overcome this limitation an exhaustive system of collecting similar trigonometric terms of intermediate expressions in the computation (routines TRIGCRUNCH and COLLECTTERMS) was implemented. Having such algebraic reduction was found to be very time consuming, so limiting conditions were placed upon when collection of terms was performed. (The user-definable parameters CTLENGTUNE and TCTUNE can be used to narrow the conditions under which term collection is performed, although in a statically defined way.) Presumably a more sophisticated LIEPROC would simplify only at appropriate times, on-the-fly.

Another way to reduce space requirements in the computation is to use a more compact representation of the class of expressions found in LIEPROC computations. In [7], Gustavson describes a Hamiltonian problem where special encoding of multivariate polynomials of pre-determined degree and number of variables enabled the rapid calculation of symbolic integrals involving polynomials up to degree eight. However, this approach can be exploited only if enough is known about the kind of expressions to develop an efficient general encoding scheme. We could not use Gustavson's scheme nor the built-in MACSYMA facilities for handling multivariate Poisson series (see Fateman [8]) because they were too limited for the generality of our problem (non-polynomial terms, possibly more than six variables). We feel that the effort involved in developing such a special representation is not

worthwhile at this stage of the development of Lie Transform-based programs even assuming that the class of Hamiltonians of interest would make this feasible and attractive.

Despite the possibilities of greater space and time efficiencies within the current context, the intrinsic intermediate expression swell (and for the coordinate transformations, final expression swell) in computations such as the example mean that the current MACSYMA memory limits will be exceeded in any case after a few more orders beyond that already possible. The alternatives for solving this problem are at the next level of difficulty - implementing a means by which parts of expressions can be placed secondary storage while not needed, or to implement a larger address space for MACSYMA on the present or an alternative model of computer (such as the DEC VAX, or the produce of MIT's LISP machine project [10]. Both require major system alterations. Nevertheless, removal of the present roadblocks is not technically infeasible, and it should be possible to carry the LIEPROC computation to higher orders in the near future.

The program can be readily extended to handle all the techniques described by McNamara [4] and to handle the several preliminary steps of canonical transformation usually required to put a Hamiltonian system into standard form. The program is very general, and therefore not optimal for the simplest cases, and only reaches fourth-order before MACSYMA runs out of memory. This is more than adequate in plasma physics applications and the program could be tuned to the special features of particular examples to go to much higher order. The super convergent transforms [4] have not been implemented and would be another way of getting to very high order - 128 in six transforms.

ACKNOWLEDGEMENTS

We wish to acknowledge the invaluable assistance of Richard Fateman, and the members of the MACSYMA group at MIT, for their generosity in offering both technical and critical advice on this project.

References

- 1 Deprit, Andre, "Canonical Transformations Depending on a Small Parameter," Celestial Mechanics, 1:12-30 (1969).
- 2 Nayfeh, A. H., Perturbation Methods, John Wiley: New York 1973. pp. 200-205.
- 3 McNamara, B., Char, B., Fateman, R., Bull. Am. Phys. Soc., p. 1122 October 1977.
- 4 McNamara, B., "Super Convergent Adiabatic Invariants with Resonant Denominators by Lie Transforms," J. Math. Phys. 2154-2164, 19 (10), 1978.
- 5 Hearn, A. C., Reduce 2 Users Manual, Second Ed. University of Utah Computation Physical Group Report No UCP-19, March, 1973.
- 6 A complete description of the facilities available in MACSYMA can be found in the MACSYMA Reference Manual, Version 9 (Massachusetts Institute of Technology Laboratory for Computer Science: Cambridge, MA, 1977). The MACSYMA Primer (MIT LCS: Cambridge, MA, 1977) is an introduction to MACSYMA usage. See also Bers, A., et al. "Symbolic Computation to Nonlinear Wave Interactions on MACSYMA," Computer Physics Communications 12:81-98 (1976).
- 7 Gustavson, F. G., "On Constructing Formal Integrals of a Hamiltonian System Near an Equilibrium Point," The Astronomical Journal, 71:8, pp. 670-686, (Oct. 1966).
- 8 Fateman, Richard J., "On the Multiplication of Poisson Series," Celestial Mechanics, 10:243-247 (1974).
- 9 McNamara, B., and Whiteman, K., "Invariants of Nearly Periodic Hamiltonian Systems," J. Math Phys. 2029-2038, 8, (10) 1967.

References Continued

- 10 Greenblatt, R., LISP Machine Progress Report Memo 444, A.I. Lab., MIT, Cambridge, Mass., August 1977.
- 11 McNamara, B., Single Particle Behaviour in Plasmas 5-25, Theoretical and Computational Plasma Physics, IAEA, Vienna, 1978.

Appendix A
USAGE OF PROCEDURE LIEPROC

1. Login in to the Macsyma system (via the ARPA network*). Then start up a Macsyma by typing: MACSYMA

Example:

```

9d 236
TRYING...
OPEN
MACSYMA CONSORTIUM KL-10

MC ITS.1126. PWORLD.1612.
TTY 53
18. LUSERS, FAIR SHARE = 68%

♦:LOGIN CHAR
PASSWORD:
[OK]
FRIDAY, SEPTEMBER 29, 1978 02:41:07 PM
WELCOME TO ITS
MILLER BACALA PAULP LINC RZ SATAN ELLEN SM DJT
CARL LINCO WSMR SASW AMO WSMR TUCKER CHAR RM
YUE

♦:MACSYM.

THIS IS MACSYMA 273

(C1)

```

2. Load the programs for the lie transform by typing
"batch (lietes ,?>, dsk, bwcucb) \$"
After several seconds, teletype will respond,
/*End of input*/
3. Then enter three parameters necessary as input for procedure lieproc.
They are:

*connection to the ARPA network from LLL can be through LLL-UNIX on site, or via the Ames TIP.

The Hamiltonian Function (H)

The Hamiltonian should be of the form

$$H(x,y) := \sum_{i=0}^k \Omega_i(x,y) \frac{\epsilon^i}{i!}$$

x and y may be names of vectors (but if so, must be of the same dimension). Each Ω_i is a scalar function. K must be finite.

Example A

The user enters a one line definition of a Hamiltonian:

(C100) $H(p,q) = p[1] + \cos(\alpha * q[1] + q[2]) * \epsilon$ \$

Here $\Omega_0(p,q) = p_1$ (first component of p)

$\Omega_1(p,q) = \cos(\alpha * q_1 + q_2)$

$\Omega_i(p,q) = 0$, for $i > 1$

Example B

The user enters a multi-line definition of a Hamiltonian, defining an array of functions, omega, which are used in the definition of the Hamiltonian:

(C100) $\text{omega}[i](u,v) := (u[1] + u[2]) * \cos(i * (v[1] + v[2]))$
 $+ \alpha * v[3]$ \$

(C101) $G(u,v) := u[1] + \epsilon * \text{omega}[1](u,v)$
 $+ \epsilon^3 * \text{omega}[3](u,v) / 3!$ \$

Here $\Omega_0(u,v) = u_1$

$\Omega_1(u,v) = \text{omega}[1](u,v) = \cos(v_1 + v_2 + \alpha v_3)(u_1 + u_2)$

$$\Omega_2 = 0$$

$$\begin{aligned}\Omega_3(u,v) &= \text{omega}[3](u,v) \\ &= (u_1 + u_2) \cos(3(v_1 + v_2) + \alpha v_3)\end{aligned}$$

Description of domain and range spaces for the transform (origin) and target). The domain and range spaces each are described as a list of the form

[vectornamel, dimension, vectorname2, dimension]

Example C

The user enters:

(C102) origin: [LILP2,LILQ,2] \$

(C103) target: [BIGP,2,BIGQ,2] \$

After entering these lines, the user can refer to the lists as "origin" and "target" respectively. Due to the nature of the lie transform, all vectors in origin and target spaces must have the same dimension. The dimensions entered must be numbers or variables with a numeric value, not symbolic constants.

4. The lie transform procedure LIEPROC can now be called, giving as parameters to it:

- a) Name of Hamiltonian
- b) How many orders to which the transformation should be computed
- c) The name of the domain space of the transform
- d) The name of the range space of the transform
- e) The name of the small parameter

Example D

The user enters:

(C104) lieproc(H,2,origin,target,epsilon) \$

after entering lines from Examples A and C above.

5. The output of lieproc

Output is then generated, order by order, starting with order 1 (K_0 is Ω_0 , by definition):

1. a) S_i , the i -th order term of the generation function S in the form of $ii(\text{function})$. The ii operator is then evaluated, and
b) the result of the evaluation (a function) is output
2. a) K_i , the i -th order term of the transformed Hamiltonian in the form of av (function).
b) The result of evaluating av ,
3. q^i : an expression indicating q as a function of P and Q (i -th order inverse transform).
4. Q^i : an expression indicating Q as a function of p and q (i -th order transform).
5. p^i : p as a function of P and Q (i -th order inverse transform).
6. P^i : P as a function of p and q (i -th order transform).

Thus,

$$q = Q + \sum_{i=1}^{\infty} \frac{\varepsilon^i}{i!} q^i (P, Q)$$

$$p = P + \sum_{i=1}^{\infty} \frac{\varepsilon^i}{i!} p^i (P, Q)$$

$$Q = q + \sum_{i=1}^{\infty} \frac{\varepsilon^i}{i!} Q^i (p, q)$$

$$P = p + \sum_{i=1}^{\infty} \frac{\varepsilon^i}{i!} P^i (p, q)$$

$$S(x, y, \varepsilon) = \sum_{i=1}^{\infty} \frac{\varepsilon^i}{i!} S_i(x, y)$$

$$K(x, y, \varepsilon) = \Omega_0(x, y) + \sum_{i=1}^{\infty} \frac{\varepsilon^i}{i!} K_i(x, y)$$

6. To leave Macsyma, and/or log out

- a) To leave Macsyma one should type quit ()\$.
- b) To leave Macsyma and logout of the system, type:
 - i) control-z (echoed as ^ z)
 - ii) :logout (CR)
 - iii) @C (CR)

Error Diagnostics

Error messages indicating Macsyma's inability to properly handle a parameter that was input may mean that the parameter was ill-defined. If such an error occurs, a fresh Macsyma can be loaded by

- a) typing :quit () \$
- b) typing :Macsyma (CR) again
- c) loading the lie programs again (see step 2)

Messages such as "You have run out of LIST space, do you want more? Type ALL; NONE; or a level no." should be responded to by typing ALL; They indicate that the Macsyma system wants to allocate more memory towards one segment of the problem.

Messages such as "LIST storage capacity exceeded," or "CORE OUT" indicate that the size of expressions involved in the computation have gotten so large that the program has run out of all available memory. Unfortunately, there is little to be done about this, and it must be regarded as a fatal error.

NOTICE

"This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights."

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

Appendix B Sample Output of Lieproc

17

```
(C1) BATCH(LIEINP,???)$  
  
(C2) /*♦PROGRAMS OF LIE TRANSFORM PROCEDURE FOR HAMILTONIANS,  
B.W. CHAR AUGUST 8, 1977  
♦/  
  
L1: [[LIEPRO,?>>], [DSK,BMCUCB], [EDITRU,?>>], [VECOPS,?>>], [LIESIM,?>>], [OPEVAL,?>>]  
[FUNCEV,?>>], [CRUNCH,?>>], [LITNOM,?>>]]$  
  
(C3) TTYOFF:TRUE$  
  
(C116) /*END OF INPUT ♦/  
  
  
(C118) H(P,Q):=(P[1]+P[2])♦(1-cos(2♦(Q[2]+Q[1])♦ALPHA))$
```

$K \text{ IS } AV(\text{LAMBDA}((P, Q), -\frac{(P_1 + P_2) \sin(Q_2 - Q_1)}{2},$
 $\frac{(1 - \cos(2(Q_2 + Q_1) \alpha))}{2}))$

$EKU (P_1, Q_1) := 0$

$LILQ^1 (BP, PQ) :=$
 $(E5) \left[-\frac{\cos(2BQ_2 \alpha + 2BQ_1 \alpha + BQ_2 - BQ_1)}{2(2\alpha - 1)}$
 $+ \frac{\cos(2BQ_2 \alpha + 2BQ_1 \alpha - BQ_2 + BQ_1)}{2(2\alpha + 1)} - \cos(BQ_2 - BQ_1),$
 $- \frac{\cos(2BQ_2 \alpha + 2BQ_1 \alpha + BQ_2 - BQ_1)}{2(2\alpha - 1)}$
 $+ \frac{\cos(2BQ_2 \alpha + 2BQ_1 \alpha - BQ_2 + BQ_1)}{2(2\alpha + 1)} - \cos(BQ_2 - BQ_1)\right]$

$BQ^1 (LILP, LILQ) :=$
 $(E6) \left[-\frac{\cos(2LILQ_2 \alpha + 2LILQ_1 \alpha + LILQ_2 - LILQ_1)}{2(2\alpha - 1)}$
 $- \frac{\cos(2LILQ_2 \alpha + 2LILQ_1 \alpha - LILQ_2 + LILQ_1)}{2(2\alpha + 1)} + \cos(LILQ_2 - LILQ_1),$
 $- \frac{\cos(2LILQ_2 \alpha + 2LILQ_1 \alpha + LILQ_2 - LILQ_1)}{2(2\alpha - 1)}$
 $- \frac{\cos(2LILQ_2 \alpha + 2LILQ_1 \alpha - LILQ_2 + LILQ_1)}{2(2\alpha + 1)} + \cos(LILQ_2 - LILQ_1)\right]$

```

(C120) BATCH(HAMIL4,>)$

(C121) /*TEST HAMILTONIAN */
H(P,Q):=P[1]+EP*OMEGA(P,Q)$

(C122) OMEGA(P,Q):=(P[1]+P[2])*(1-COS(2*ALPHA*(Q[1]+Q[2])))
           *SIN((Q[1]-Q[2]))$
$ 

(C123) /*DESCRIPTIONS OF ORIGIN AND DESTINATION SPACES*/
ORIGIN:{LILP,2,LILQ,2}$

(C124) TARGET:{BP,2,BQ,2}$

(C125) /*PARAMETERS CONTROLLING THE AMOUNT OF TRIGONOMETRIC TERM COLLECTION*/
CTLENGTUNE:40$

(C126) TCTUNE:3$

(C128) LIEPROC(H,2,ORIGIN,TARGET,EP)$
{LILP, LILQ} {BP, BQ}
ORDER 1 TERMS ARE--
S IS   II(LAMBDA([P, Q], - (P + P ) SIN(Q - Q )
           2      1            2      1
                                         (1 - COS(2 (Q + Q ) ALPHA)))
           2      1

ISOLAT FASL DSK MAXOUT being loaded
Loading done

TRGRED FASL DSK MACSYM being loaded
Loading done

SCHATC FASL DSK MACSYM being loaded
Loading done

STATUS FASL DSK MAXOUT being loaded
Loading done
ESU ( P , Q ):=
 1
      (P + P ) COS(2 Q ALPHA + 2 Q ALPHA + Q - Q )
      2      1            2      1            2      1
(E3) - -----
                  2 (2 ALPHA - 1)
      (P + P ) COS(2 Q ALPHA + 2 Q ALPHA - Q + Q )
      2      1            2      1            2      1
+ ----- - (P + P ) COS(Q - Q )
      2 (2 ALPHA + 1)            2      1            2      1

```

ORDER 2 TERMS ARE--

BINOMIAL FASL DSK MANOUT being loaded
Loading done

PRODUCT FASL DSK MAXOUT being loaded

Loading done
S IS II(LAMBDA([P, Q],

$$\begin{aligned} & \frac{2 (P_2 + P_1) \alpha \cos(2Q_2 \alpha + 2Q_1 \alpha + 2Q_2 - 2Q_1)}{2 \alpha - 1} \\ & + \frac{2 (P_2 + P_1) \alpha \cos(2Q_2 \alpha + 2Q_1 \alpha - 2Q_2 + 2Q_1)}{2 \alpha + 1} \\ & + \frac{4 (P_2 + P_1) \alpha \cos(2Q_2 \alpha + 2Q_1 \alpha)}{(2 \alpha - 1) (2 \alpha + 1)} \\ & + \frac{4 (P_2 + P_1) \cos(2Q_2 - 2Q_1) \alpha^2}{(2 \alpha - 1) (2 \alpha + 1)} - \frac{4 (P_2 + P_1) \alpha^2}{(2 \alpha - 1) (2 \alpha + 1)} \end{aligned}$$

ESU (P, Q) :=

$$\begin{aligned} & \frac{(P_2 + P_1) \alpha \sin(2Q_2 \alpha + 2Q_1 \alpha + 2Q_2 - 2Q_1)}{(\alpha - 1) (2 \alpha - 1)} \\ & + \frac{(P_2 + P_1) \alpha \sin(2Q_2 \alpha + 2Q_1 \alpha - 2Q_2 + 2Q_1)}{(\alpha + 1) (2 \alpha + 1)} \\ & + \frac{2 (P_2 + P_1) \alpha \sin(2Q_2 \alpha + 2Q_1 \alpha)}{(2 \alpha - 1) (2 \alpha + 1)} \\ & - \frac{2 (P_2 + P_1) \sin(2Q_2 - 2Q_1) \alpha^2}{(2 \alpha - 1) (2 \alpha + 1)} \end{aligned}$$

LILP (BP, BQ) :=

$$\begin{aligned} & \frac{(BP_2 + BP_1) \sin(2BQ_2 \alpha + 2BQ_1 \alpha + BQ_2 - BQ_1)}{2} \\ & (E7) [- \frac{(BP_2 + BP_1) \sin(2BQ_2 \alpha + 2BQ_1 \alpha - BQ_2 + BQ_1)}{2} \\ & + \frac{(BP_2 + BP_1) \sin(BQ_2 - BQ_1)}{2} \\ & + \frac{(BP_2 + BP_1) (2 \alpha + 1) \sin(2BQ_2 \alpha + 2BQ_1 \alpha + BQ_2 - BQ_1)}{2 (2 \alpha - 1)} \\ & + \frac{(BP_2 + BP_1) (2 \alpha - 1) \sin(2BQ_2 \alpha + 2BQ_1 \alpha - BQ_2 + BQ_1)}{2 (2 \alpha + 1)} \\ & - (BP_2 + BP_1) \sin(BQ_2 - BQ_1)] \end{aligned}$$

BP¹ (LILP, LILQ) :=

$$\begin{aligned} & \frac{(LILP_2 + LILP_1) \sin(2LILQ_2 \alpha + 2LILQ_1 \alpha + LILQ_2 - LILQ_1)}{2} \\ & (E8) [- \frac{(LILP_2 + LILP_1) \sin(2LILQ_2 \alpha + 2LILQ_1 \alpha - LILQ_2 + LILQ_1)}{2} \\ & - (LILP_2 + LILP_1) \sin(LILQ_2 - LILQ_1) \\ & (LILP_2 + LILP_1) (2 \alpha + 1) \sin(2LILQ_2 \alpha + 2LILQ_1 \alpha + LILQ_2 - LILQ_1) \\ & - LILQ_1 / (2 (2 \alpha - 1)) - (LILP_2 + LILP_1) (2 \alpha - 1) \\ & \sin(2LILQ_2 \alpha + 2LILQ_1 \alpha - LILQ_2 + LILQ_1) / (2 (2 \alpha + 1)) \\ & + (LILP_2 + LILP_1) \sin(LILQ_2 - LILQ_1) \end{aligned}$$

$$\text{LILQ} (\text{BP}, \text{BQ}) :=$$

$$\frac{\sin(4 \text{BQ}_2 \text{ALPHA} + 4 \text{BQ}_1 \text{ALPHA} + 2 \text{BQ}_2 - 2 \text{BQ}_1)}{4 (2 \text{ALPHA} - 1)}$$

$$- \frac{\sin(4 \text{BQ}_2 \text{ALPHA} + 4 \text{BQ}_1 \text{ALPHA} - 2 \text{BQ}_2 + 2 \text{BQ}_1)}{4 (2 \text{ALPHA} + 1)}$$

$$+ \frac{\text{ALPHA} \sin(4 \text{BQ}_2 \text{ALPHA} + 4 \text{BQ}_1 \text{ALPHA})}{(2 \text{ALPHA} - 1) (2 \text{ALPHA} + 1)}$$

$$- \frac{(2 \text{ALPHA}^2 - 2 \text{ALPHA} + 1) \sin(2 \text{BQ}_2 \text{ALPHA} + 2 \text{BQ}_1 \text{ALPHA} + 2 \text{BQ}_2 - 2 \text{BQ}_1)}{(\text{ALPHA} - 1) (2 \text{ALPHA} - 1)}$$

$$+ \frac{(2 \text{ALPHA}^2 + 2 \text{ALPHA} + 1) \sin(2 \text{BQ}_2 \text{ALPHA} + 2 \text{BQ}_1 \text{ALPHA} - 2 \text{BQ}_2 + 2 \text{BQ}_1)}{(\text{ALPHA} + 1) (2 \text{ALPHA} + 1)}$$

$$+ \frac{\sin(2 \text{BQ}_2 - 2 \text{BQ}_1) (4 \text{ALPHA}^2 - 3)}{2 (2 \text{ALPHA} - 1) (2 \text{ALPHA} + 1)}$$

$$- \frac{(2 \text{ALPHA} + 1) \sin(4 \text{BQ}_2 \text{ALPHA} + 4 \text{BQ}_1 \text{ALPHA} + 2 \text{BQ}_2 - 2 \text{BQ}_1)}{4 (2 \text{ALPHA} - 1)}$$

$$- \frac{(2 \text{ALPHA} - 1) \sin(4 \text{BQ}_2 \text{ALPHA} + 4 \text{BQ}_1 \text{ALPHA} - 2 \text{BQ}_2 + 2 \text{BQ}_1)}{4 (2 \text{ALPHA} + 1)}$$

$$+ \frac{\text{ALPHA} \sin(4 \text{BQ}_2 \text{ALPHA} + 4 \text{BQ}_1 \text{ALPHA})}{(2 \text{ALPHA} - 1) (2 \text{ALPHA} + 1)}$$

$$- \frac{(2 \text{ALPHA}^2 - 1) \sin(2 \text{BQ}_2 \text{ALPHA} + 2 \text{BQ}_1 \text{ALPHA} + 2 \text{BQ}_2 - 2 \text{BQ}_1)}{(\text{ALPHA} - 1) (2 \text{ALPHA} - 1)}$$

$$\text{K IS } \text{AV}(\text{LAMBDA}([P, Q]),$$

$$- \frac{2 (P_2^2 + P_1^2) \text{ALPHA}^2 \cos(2 Q_2 \text{ALPHA} + 2 Q_1 \text{ALPHA} + 2 Q_2 - 2 Q_1)}{2 \text{ALPHA} - 1}$$

$$+ \frac{2 (P_2^2 + P_1^2) \text{ALPHA}^2 \cos(2 Q_2 \text{ALPHA} + 2 Q_1 \text{ALPHA} - 2 Q_2 + 2 Q_1)}{2 \text{ALPHA} + 1}$$

$$+ \frac{4 (P_2^2 + P_1^2) \text{ALPHA}^2 \cos(2 Q_2 \text{ALPHA} + 2 Q_1 \text{ALPHA})}{(2 \text{ALPHA} - 1) (2 \text{ALPHA} + 1)}$$

$$+ \frac{4 (P_2^2 + P_1^2) \cos(2 Q_2 - 2 Q_1) \text{ALPHA}^2}{(2 \text{ALPHA} - 1) (2 \text{ALPHA} + 1)} - \frac{4 (P_2^2 + P_1^2) \text{ALPHA}^2}{(2 \text{ALPHA} - 1) (2 \text{ALPHA} + 1))}$$

$$\text{EKU} (\text{P}, \text{Q}) :=$$

$$2$$

$$(E10)$$

$$- \frac{4 (P_2^2 + P_1^2) \text{ALPHA}^2}{(2 \text{ALPHA} - 1) (2 \text{ALPHA} + 1)}$$

$$\begin{aligned}
& \frac{(2 \alpha - 1) \sin(2 \beta_0 \alpha + 2 \beta_0 \alpha - 2 \beta_0 + 2 \beta_0)}{(\alpha + 1)(2 \alpha + 1)} \\
& + \frac{3 \sin(2 \beta_0 - 2 \beta_0)}{2} \\
& \frac{\beta_0^2 (\text{LILP}, \text{LILC}) :=}{\text{SIN}(4 \text{LILQ}_2 \alpha + 4 \text{LILQ}_1 \alpha + 2 \text{LILQ}_2 - 2 \text{LILQ}_1)} \\
& (E12) \{ 2 (- \frac{4 (2 \alpha - 1)}{\text{SIN}(4 \text{LILQ}_2 \alpha + 4 \text{LILQ}_1 \alpha - 2 \text{LILQ}_2 + 2 \text{LILQ}_1)} \\
& - \frac{4 (2 \alpha + 1)}{\text{ALPHA SIN}(4 \text{LILQ}_2 \alpha + 4 \text{LILQ}_1 \alpha)} \\
& + \frac{(2 \alpha - 1) \text{SIN}(2 \text{LILC}_2 \alpha + 2 \text{LILQ}_1 \alpha + 2 \text{LILQ}_2 - 2 \text{LILQ}_1)}{2 \alpha - 1} \\
& - \frac{(2 \alpha + 1) \text{SIN}(2 \text{LILQ}_2 \alpha + 2 \text{LILQ}_1 \alpha - 2 \text{LILQ}_2 + 2 \text{LILQ}_1)}{2 \alpha + 1} \\
& - \frac{2 \alpha \text{SIN}(2 \text{LILQ}_2 \alpha + 2 \text{LILQ}_1 \alpha)}{(2 \alpha - 1)(2 \alpha + 1)} \\
& + \frac{\text{SIN}(2 \text{LILQ}_2 - 2 \text{LILQ}_1) (8 \alpha^2 - 3)}{2 (2 \alpha - 1)(2 \alpha + 1)} \\
& - \frac{\text{SIN}(4 \text{LILQ}_2 \alpha + 4 \text{LILQ}_1 \alpha + 2 \text{LILQ}_2 - 2 \text{LILQ}_1)}{4 (2 \alpha - 1)}
\end{aligned}$$

$$\begin{aligned}
& \frac{\text{SIN}(4 \text{LILQ}_2 \alpha + 4 \text{LILQ}_1 \alpha - 2 \text{LILQ}_2 + 2 \text{LILQ}_1)}{4 (2 \alpha + 1)} \\
& - \frac{\text{ALPHA SIN}(4 \text{LILQ}_2 \alpha + 4 \text{LILQ}_1 \alpha)}{(2 \alpha - 1)(2 \alpha + 1)} \\
& + \frac{(2 \alpha - 2 \alpha + 1) \text{SIN}(2 \text{LILQ}_2 \alpha + 2 \text{LILQ}_1 \alpha + 2 \text{LILQ}_2 - 2 \text{LILQ}_1)}{(1 - 2 \text{LILQ}_1) / ((\alpha - 1)(2 \alpha - 1))} \\
& - \frac{(2 \alpha + 2 \alpha + 1) \text{SIN}(2 \text{LILQ}_2 \alpha + 2 \text{LILQ}_1 \alpha - 2 \text{LILQ}_2 + 2 \text{LILQ}_1)}{(1 - 2 \text{LILQ}_1) / ((\alpha + 1)(2 \alpha + 1))} \\
& - \frac{\text{SIN}(2 \text{LILQ}_2 - 2 \text{LILQ}_1) (4 \alpha^2 - 3)}{2 (2 \alpha - 1)(2 \alpha + 1)} \\
& - \frac{(2 \alpha + 1) \text{SIN}(4 \text{LILQ}_2 \alpha + 4 \text{LILQ}_1 \alpha + 2 \text{LILQ}_2 - 2 \text{LILQ}_1)}{4 (2 \alpha - 1)} \\
& - \frac{(2 \alpha - 1) \text{SIN}(4 \text{LILQ}_2 \alpha + 4 \text{LILQ}_1 \alpha - 2 \text{LILQ}_2 + 2 \text{LILQ}_1)}{4 (2 \alpha + 1)} \\
& - \frac{\text{ALPHA SIN}(4 \text{LILQ}_2 \alpha + 4 \text{LILQ}_1 \alpha)}{(2 \alpha - 1)(2 \alpha + 1)} \\
& - \frac{(2 \alpha + 1) \text{SIN}(2 \text{LILQ}_2 \alpha + 2 \text{LILQ}_1 \alpha + 2 \text{LILQ}_2 - 2 \text{LILQ}_1)}{2 \alpha - 1} \\
& - \frac{(2 \alpha - 1) \text{SIN}(2 \text{LILQ}_2 \alpha + 2 \text{LILQ}_1 \alpha - 2 \text{LILQ}_2 + 2 \text{LILQ}_1)}{2 \alpha + 1} \\
& - \frac{2 \alpha \text{SIN}(2 \text{LILQ}_2 \alpha + 2 \text{LILQ}_1 \alpha)}{(2 \alpha - 1)(2 \alpha + 1)}
\end{aligned}$$

$$\begin{aligned}
& \frac{\sin(2 \cdot \text{LILQ}_2 - 2 \cdot \text{LILQ}_1) (8 \cdot \text{ALPHA}^2 - 3)}{2 \cdot 1} \\
& - \frac{2 \cdot (2 \cdot \text{ALPHA} - 1) (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& + \frac{(2 \cdot \text{ALPHA} + 1) \sin(4 \cdot \text{LILQ}_2 \cdot \text{ALPHA} + 4 \cdot \text{LILQ}_1 \cdot \text{ALPHA} + 2 \cdot \text{LILQ}_2 - 2 \cdot \text{LILQ}_1)}{2 \cdot 1} \\
& + \frac{4 \cdot (2 \cdot \text{ALPHA} - 1)}{2 \cdot 1} \\
& + \frac{(2 \cdot \text{ALPHA} - 1) \sin(4 \cdot \text{LILQ}_2 \cdot \text{ALPHA} + 4 \cdot \text{LILQ}_1 \cdot \text{ALPHA} - 2 \cdot \text{LILQ}_2 + 2 \cdot \text{LILQ}_1)}{2 \cdot 1} \\
& + \frac{4 \cdot (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& - \frac{\text{ALPHA} \sin(4 \cdot \text{LILQ}_2 \cdot \text{ALPHA} + 4 \cdot \text{LILQ}_1 \cdot \text{ALPHA})}{2 \cdot 1} \\
& - \frac{(2 \cdot \text{ALPHA} - 1) (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& + \frac{(2 \cdot \text{ALPHA}^2 - 1) \sin(2 \cdot \text{LILQ}_2 \cdot \text{ALPHA} + 2 \cdot \text{LILQ}_1 \cdot \text{ALPHA} + 2 \cdot \text{LILQ}_2 - 2 \cdot \text{LILQ}_1)}{2 \cdot 1} \\
& + \frac{(\text{ALPHA} - 1) (2 \cdot \text{ALPHA} - 1)}{2 \cdot 1} \\
& - \frac{(2 \cdot \text{ALPHA}^2 - 1) \sin(2 \cdot \text{LILQ}_2 \cdot \text{ALPHA} + 2 \cdot \text{LILQ}_1 \cdot \text{ALPHA} - 2 \cdot \text{LILQ}_2 + 2 \cdot \text{LILQ}_1)}{2 \cdot 1} \\
& - \frac{(\text{ALPHA} + 1) (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& + \frac{3 \sin(2 \cdot \text{LILQ}_2 - 2 \cdot \text{LILQ}_1)}{2} \\
& + \frac{2}{2}
\end{aligned}$$

LILP (BP, BQ) :=

(E13)

$$\begin{aligned}
& \frac{(BP_2 + BP_1) (2 \cdot \text{ALPHA} + 1) \cos(4 \cdot BQ_2 \cdot \text{ALPHA} + 4 \cdot BQ_1 \cdot \text{ALPHA} + 2 \cdot BQ_2 - 2 \cdot BQ_1)}{2 \cdot 1} \\
& - \frac{4 \cdot (2 \cdot \text{ALPHA} - 1)}{2 \cdot 1} \\
& + \frac{(BP_2 + BP_1) (2 \cdot \text{ALPHA} - 1) \cos(4 \cdot BQ_2 \cdot \text{ALPHA} + 4 \cdot BQ_1 \cdot \text{ALPHA} - 2 \cdot BQ_2 + 2 \cdot BQ_1)}{2 \cdot 1} \\
& + \frac{4 \cdot (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& - \frac{(BP_2 + BP_1) \cos(4 \cdot BQ_2 \cdot \text{ALPHA} + 4 \cdot BQ_1 \cdot \text{ALPHA})}{2 \cdot 1} \\
& - \frac{2 \cdot (2 \cdot \text{ALPHA} - 1) (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1}
\end{aligned}$$

$$\begin{aligned}
& + \frac{(BP_2 + BP_1) (4 \cdot \text{ALPHA}^2 - \text{ALPHA} - 1) \cos(2 \cdot BQ_2 \cdot \text{ALPHA} + 2 \cdot BQ_1 \cdot \text{ALPHA} + 2 \cdot BQ_2)}{2 \cdot 1} \\
& - \frac{2 \cdot BQ_1 / (2 \cdot \text{ALPHA} - 1) - (BP_2 + BP_1) (4 \cdot \text{ALPHA}^2 + \text{ALPHA} - 1)}{2 \cdot 1} \\
& \cos(2 \cdot BQ_2 \cdot \text{ALPHA} + 2 \cdot BQ_1 \cdot \text{ALPHA} - 2 \cdot BQ_2 + 2 \cdot BQ_1) / (2 \cdot \text{ALPHA} + 1) \\
& - \frac{2 \cdot (BP_2 + BP_1) (2 \cdot \text{ALPHA}^2 - 1) \cos(2 \cdot BQ_2 \cdot \text{ALPHA} + 2 \cdot BQ_1 \cdot \text{ALPHA})}{2 \cdot 1} \\
& - \frac{(2 \cdot \text{ALPHA} - 1) (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& + \frac{(BP_2 + BP_1) (64 \cdot \text{ALPHA}^4 - 12 \cdot \text{ALPHA}^2 + 3)}{2 \cdot 1} \\
& - \frac{2 \cdot (2 \cdot \text{ALPHA} - 1) (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& - \frac{3 (EP_2 + EP_1) \cos(2 \cdot BQ_2 - 2 \cdot BQ_1) (8 \cdot \text{ALPHA}^2 - 1)}{2 \cdot 1} \\
& - \frac{2 \cdot (2 \cdot \text{ALPHA} - 1) (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& - \frac{(BP_2 + BP_1) \cos(4 \cdot BQ_2 \cdot \text{ALPHA} + 4 \cdot BQ_1 \cdot \text{ALPHA} + 2 \cdot BQ_2 - 2 \cdot BQ_1)}{2 \cdot 1} \\
& - \frac{4 \cdot (2 \cdot \text{ALPHA} - 1)}{2 \cdot 1} \\
& - \frac{(BP_2 + BP_1) \cos(4 \cdot BQ_2 \cdot \text{ALPHA} + 4 \cdot BQ_1 \cdot \text{ALPHA} - 2 \cdot BQ_2 + 2 \cdot BQ_1)}{2 \cdot 1} \\
& - \frac{4 \cdot (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& - \frac{(BP_2 + BP_1) \cos(4 \cdot BQ_2 \cdot \text{ALPHA} + 4 \cdot BQ_1 \cdot \text{ALPHA})}{2 \cdot 1} \\
& - \frac{2 \cdot (2 \cdot \text{ALPHA} - 1) (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& + \frac{(BP_2 + BP_1) (\text{ALPHA} + 1) (4 \cdot \text{ALPHA}^2 - 3 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& \cos(2 \cdot BQ_2 \cdot \text{ALPHA} + 2 \cdot BQ_1 \cdot \text{ALPHA} + 2 \cdot BQ_2 - 2 \cdot BQ_1) / ((\text{ALPHA} - 1) (2 \cdot \text{ALPHA} - 1)) \\
& - \frac{(BP_2 + BP_1) (\text{ALPHA} - 1) (4 \cdot \text{ALPHA}^2 + 3 \cdot \text{ALPHA} + 1)}{2 \cdot 1} \\
& \cos(2 \cdot BQ_2 \cdot \text{ALPHA} + 2 \cdot BQ_1 \cdot \text{ALPHA} - 2 \cdot BQ_2 + 2 \cdot BQ_1) / ((\text{ALPHA} + 1) (2 \cdot \text{ALPHA} + 1)) \\
& - \frac{2 \cdot (BP_2 + BP_1) (2 \cdot \text{ALPHA} + 1) \cos(2 \cdot BQ_2 \cdot \text{ALPHA} + 2 \cdot BQ_1 \cdot \text{ALPHA})}{2 \cdot 1} \\
& + \frac{(2 \cdot \text{ALPHA} - 1) (2 \cdot \text{ALPHA} + 1)}{2 \cdot 1}
\end{aligned}$$

$$\begin{aligned}
& \frac{(BP_2 + BP_1) (64 \alpha^4 + 4 \alpha^2 + 3)}{2 (2 \alpha - 1) (2 \alpha + 1)} \\
& + \frac{(BP_2 + BP_1) \cos(2 BQ_2 - 2 BO_1) (8 \alpha^2 - 3)}{2 (2 \alpha - 1) (2 \alpha + 1)} \\
& \frac{BP_2 (LILP_2, LILQ_1) :=}{(E14) [2 (- (LILP_2 + LILP_1) (2 \alpha + 1) \\
& \cos(4 LILQ_2 \alpha^2 + 4 LILQ_1 \alpha^1 + 2 LILQ_2 \alpha^0 - 2 LILQ_1 \alpha^{-1}) / (4 (2 \alpha - 1)^2) \\
& + (LILP_2 + LILP_1) (2 \alpha - 1) \cos(4 LILQ_2 \alpha^2 + 4 LILQ_1 \alpha^1 - 2 LILQ_2 \alpha^0 \\
& + 2 LILQ_1 \alpha^{-1}) / (4 (2 \alpha + 1)^2) \\
& (LILP_2 + LILP_1) \cos(4 LILQ_2 \alpha^2 + 4 LILQ_1 \alpha^1) \\
& - 2 (2 \alpha - 1) (2 \alpha + 1) \\
& + (LILP_2 + LILP_1) (\alpha - 1) (2 \alpha + 1) \\
& \cos(2 LILQ_2 \alpha^2 + 2 LILQ_1 \alpha^1 + 2 LILQ_2 \alpha^0 - 2 LILQ_1 \alpha^{-1}) / (2 \alpha - 1) \\
& - (LILP_2 + LILP_1) (\alpha + 1) (2 \alpha - 1) \\
& \cos(2 LILQ_2 \alpha^2 + 2 LILQ_1 \alpha^1 - 2 LILQ_2 \alpha^0 + 2 LILQ_1 \alpha^{-1}) / (2 \alpha + 1) \\
& + (LILP_2 + LILP_1) \cos(2 LILQ_2 \alpha^2 + 2 LILQ_1 \alpha^1) \\
& - 2 (2 \alpha - 1) (2 \alpha + 1) \\
& + (LILP_2 + LILP_1) (64 \alpha^4 - 12 \alpha^2 + 3) \\
& + \frac{(LILP_2 + LILP_1) (64 \alpha^4 - 12 \alpha^2 + 3)}{2 (2 \alpha - 1) (2 \alpha + 1)} \\
& \frac{(LILP_2 + LILP_1) \cos(2 LILQ_2 \alpha^2 - 2 LILQ_1 \alpha^1) (16 \alpha^2 - 3)}{2 (2 \alpha - 1) (2 \alpha + 1)} \\
& + (LILP_2 + LILP_1) (2 \alpha + 1) \cos(4 LILQ_2 \alpha^2 + 4 LILQ_1 \alpha^1 + 2 LILQ_2 \alpha^0 \\
& - 2 LILQ_1 \alpha^{-1}) / (4 (2 \alpha - 1)^2) - (LILP_2 + LILP_1) (2 \alpha - 1) \\
& \cos(4 LILQ_2 \alpha^2 + 4 LILQ_1 \alpha^1 - 2 LILQ_2 \alpha^0 + 2 LILQ_1 \alpha^{-1}) / (4 (2 \alpha + 1)^2) \\
& (LILP_2 + LILP_1) \cos(4 LILQ_2 \alpha^2 + 4 LILQ_1 \alpha^1) \\
& + \frac{(LILP_2 + LILP_1) (4 \alpha^2 - \alpha - 1)}{2 (2 \alpha - 1) (2 \alpha + 1)} \\
& - (LILP_2 + LILP_1) (4 \alpha^2 + \alpha - 1) \\
& \cos(2 LILQ_2 \alpha^2 + 2 LILQ_1 \alpha^1 + 2 LILQ_2 \alpha^0 - 2 LILQ_1 \alpha^{-1}) / (2 \alpha - 1) \\
& + (LILP_2 + LILP_1) (4 \alpha^2 + \alpha - 1) \\
& \cos(2 LILQ_2 \alpha^2 + 2 LILQ_1 \alpha^1 - 2 LILQ_2 \alpha^0 + 2 LILQ_1 \alpha^{-1}) / (2 \alpha + 1) \\
& + (LILP_2 + LILP_1) (2 \alpha - 1) \cos(2 LILQ_2 \alpha^2 + 2 LILQ_1 \alpha^1) \\
& - \frac{(LILP_2 + LILP_1) (2 \alpha - 1) (2 \alpha + 1)}{(2 \alpha - 1) (2 \alpha + 1)} \\
& \frac{(LILP_2 + LILP_1) (64 \alpha^4 - 12 \alpha^2 + 3)}{2 (2 \alpha - 1) (2 \alpha + 1)} \\
& + \frac{3 (LILP_2 + LILP_1) \cos(2 LILQ_2 \alpha^2 - 2 LILQ_1 \alpha^1) (8 \alpha^2 - 1)}{2 (2 \alpha - 1) (2 \alpha + 1)} \\
& + (LILP_2 + LILP_1) \cos(4 LILQ_2 \alpha^2 + 4 LILQ_1 \alpha^1 + 2 LILQ_2 \alpha^0 - 2 LILQ_1 \alpha^{-1}) \\
& 2 (-----) / (4 (2 \alpha - 1))
\end{aligned}$$

$$\begin{aligned}
& \frac{(4 \text{ALPHA}^2 + 3 \text{ALPHA} + 1) \cos(2 \text{LILQ}_2 \text{ALPHA} + 2 \text{LILQ}_1 \text{ALPHA} - 2 \text{LILQ}_2)}{2} \\
& + \frac{2 \text{LILQ}_1}{((\text{ALPHA} + 1)(2 \text{ALPHA} + 1))} \\
& - \frac{2 (\text{LILP}_2 + \text{LILP}_1) (2 \text{ALPHA}^2 + 1) \cos(2 \text{LILQ}_2 \text{ALPHA} + 2 \text{LILQ}_1 \text{ALPHA})}{(2 \text{ALPHA} - 1)(2 \text{ALPHA} + 1)} \\
& - \frac{(\text{LILP}_2 + \text{LILP}_1) (64 \text{ALPHA}^4 + 4 \text{ALPHA}^2 + 3)}{2 (2 \text{ALPHA} - 1)(2 \text{ALPHA} + 1)} \\
& + \frac{(\text{LILP}_2 + \text{LILP}_1) \cos(2 \text{LILQ}_2 - 2 \text{LILQ}_1) (8 \text{ALPHA}^2 - 3)}{2 (2 \text{ALPHA} - 1)(2 \text{ALPHA} + 1)}
\end{aligned}$$

$$\begin{aligned}
& \frac{(\text{LILP}_2 + \text{LILP}_1) \cos(4 \text{LILQ}_2 \text{ALPHA} + 4 \text{LILQ}_1 \text{ALPHA} - 2 \text{LILQ}_2 + 2 \text{LILQ}_1)}{4 (2 \text{ALPHA} + 1)} \\
& - \frac{(\text{LILP}_2 + \text{LILP}_1) \cos(4 \text{LILQ}_2 \text{ALPHA} + 4 \text{LILQ}_1 \text{ALPHA})}{2 (2 \text{ALPHA} - 1)(2 \text{ALPHA} + 1)} \\
& + \frac{(\text{LILP}_2 + \text{LILP}_1) (\text{ALPHA} + 1) \cos(2 \text{LILQ}_2 \text{ALPHA} + 2 \text{LILQ}_1 \text{ALPHA} + 2 \text{LILQ}_2)}{2} \\
& - 2 \text{LILQ}_1 = (\text{LILP}_2 + \text{LILP}_1) (\text{ALPHA} - 1) \\
& \cos(2 \text{LILQ}_2 \text{ALPHA} + 2 \text{LILQ}_1 \text{ALPHA} - 2 \text{LILQ}_2 + 2 \text{LILQ}_1) \\
& - \frac{2 (\text{LILP}_2 + \text{LILP}_1) (4 \text{ALPHA}^2 + 1) \cos(2 \text{LILQ}_2 \text{ALPHA} + 2 \text{LILQ}_1 \text{ALPHA})}{(2 \text{ALPHA} - 1)(2 \text{ALPHA} + 1)} \\
& + \frac{(\text{LILP}_2 + \text{LILP}_1) (64 \text{ALPHA}^4 + 4 \text{ALPHA}^2 + 3)}{2 (2 \text{ALPHA} - 1)(2 \text{ALPHA} + 1)} \\
& - \frac{(\text{LILP}_2 + \text{LILP}_1) \cos(2 \text{LILQ}_2 - 2 \text{LILQ}_1) (16 \text{ALPHA}^2 - 3)}{2 (2 \text{ALPHA} - 1)(2 \text{ALPHA} + 1)} \\
& - \frac{(\text{LILP}_2 + \text{LILP}_1) \cos(4 \text{LILQ}_2 \text{ALPHA} + 4 \text{LILQ}_1 \text{ALPHA} + 2 \text{LILQ}_2 - 2 \text{LILQ}_1)}{4 (2 \text{ALPHA} - 1)} \\
& + \frac{(\text{LILP}_2 + \text{LILP}_1) \cos(4 \text{LILQ}_2 \text{ALPHA} + 4 \text{LILQ}_1 \text{ALPHA} - 2 \text{LILQ}_2 + 2 \text{LILQ}_1)}{4 (2 \text{ALPHA} + 1)} \\
& + \frac{(\text{LILP}_2 + \text{LILP}_1) \cos(4 \text{LILQ}_2 \text{ALPHA} + 4 \text{LILQ}_1 \text{ALPHA})}{2 (2 \text{ALPHA} - 1)(2 \text{ALPHA} + 1)} \\
& - (\text{LILP}_2 + \text{LILP}_1) (\text{ALPHA} + 1) (4 \text{ALPHA}^2 - 3 \text{ALPHA} + 1) \\
& \cos(2 \text{LILQ}_2 \text{ALPHA} + 2 \text{LILQ}_1 \text{ALPHA} + 2 \text{LILQ}_2 - 2 \text{LILQ}_1) \\
& /((\text{ALPHA} - 1)(2 \text{ALPHA} - 1)) + (\text{LILP}_2 + \text{LILP}_1) (\text{ALPHA} - 1)
\end{aligned}$$

```

(C2) /*Programs of Lie Transform Procedure for Hamiltonians,
      B.W. Char August 8, 1977
*/
L1:[[LIEPRO,?>],DSK,BWCUCB],[EDITRU,?>],[VECOPS,?>],[LIESIM,?>],[OPEVAL,?>],
      [FUNCEV,?>],[CRUNCH,?>],[LITNON,?>]]$

(C3) INPUT(L):=APPLY(BATCH,L)$

(C4) MAP(INPUT,L1)$

(C5) /*The following state the basic formulas for the special case of the
lie transform having to do with canonical transformations of Hamiltonian
systems. This particular transform produces a canonical system with
Hamiltonian K free of terms involving Q[1]. The notation used is
from Nayfeh, and notes from McNamara (Mar., 1977)

      AV and II are special functional operators defined in McNamara's
notes and further documented elsewhere in this listing. PB is the
poisson bracket operator. OMU[I] is the ith term of the Hamiltonian specified as
into to Lieproc (see description of lieproc below).
*/
/* Where any confusion might result, we use the following standard notations
for translating between notes and program:
xu = uppercase x (possibly subscripted)
xl = lowercase x (possibly subscripted)
xx = doubly subscripted x (e.g. xxl[i,j])
xs = superscripted x (e.g. xls[n])
indexes such as i,j,n,m are unchanged
*/
TU[N]:=EVALLIEB2(OMU[N]+SUM(BINOMIAL(N-1,J-1)*PB(OMU[N-J],ESU[J])
      +BINOMIAL(N-1,J)*EKNU[J,N-J],J,1,N-1))$

(C6) KU[0]:=OMU[0]$

(C7) KU[N]:=AV(TU[N])$

(C8) EKU[N]:=EVALLIEB2(KU[N])$

(C9) EKNU[J,I]:=EVALLIEB2( PB(EKU[I],ESU[J])-  

      SUM(BINOMIAL(J-1,M-1)*PB(EKNU[J-M,I],ESU[I]), M,1,J-1))$

(C10) SU[N]:= II(TU[N])$

(C11) ESU[N]:=EVALLIEB2(SU[N])$

(C12) QLS[N]:=EVALLIEB2(MAP(EV,APPLY(LAMBDA,['DUMMYDOMAIN,APPLY(VDIFF,[ESU[N],P]))])+SUM(BINOMIAL(N-1,J)*QQL[J,N-J],J,1,N-1))$

(C13) QQL[J,I]:=EVALLIEB2(PB(QLS[I],ESU[J])- SUM(BINOMIAL(J-1,M-1)*PB(QQL[J-M,I],ESU[M]),M,1,J-1))$

(C14) PLS[N]:=EVALLIEB2(MAP(EV,APPLY(LAMBDA,['DUMMYDOMAIN,-APPLY(VDIFF,[ESU[N],Q]))])+SUM(BINOMIAL(N-1,J)*PPL[N-1,J],J,1,N-1))$

(C15) PPL[J,I]:=EVALLIEB2(PB(PLS[I],ESU[J])-SUM(BINOMIAL(J-1,M-1)*PB(PPL[J-M,I],ESU[M]),M,1,J-1))$

(C16) QUS[N](P,Q):=-QLS[N](P,Q)+SUM(BINOMIAL(N,J)*QQL[J,N-J](P,Q),J,1,N-1)$

(C17) PUS[N](P,Q):=-PLS[N](P,Q)+SUM(BINOMIAL(N,J)*PPL[J,N-J](P,Q),J,1,N-1)$

```

(C18) /* Lieproc is the procedure to call in order to actually get the lie transform of something computed. LIEPROC assumes that the Hamiltonian function (e.g. H(p,q):=p[1]+cos(p[2]+3*q[2])*ep+sin(p[3]+q[3])*ep^2) has already been typed in beforehand. L1 is the description of the source space--the variables to be transformed. L1 must be a series of: a vector variable name, followed by its (the vector's) length, e.g. [littlep,3,littleq,3] would define the source space to be six dimensional, the first three components referred to as littlep[1], littlep[2], and littlep[3], the second three similarly.

L2 must contain a description of the destination space, in a format similar to that of L1. For example, [bigp,3,bigq,3] could be a description of a six dimensional space .

The value of N determines to how many orders the transform shall be computed. EP should be the name of the small parameter used in the Hamiltonian (Referred to as epsilon in most descriptions of the Lie Transform).

Once called, LIEPROC will compute and print out, for a given order: S (generating function), K, q-super-i and p-super-i (the i-th term of the transformation from the destination space to the source space--the "inverse transform"), and Q-super-i and P-super-i (the transform from source to destination space). S and K are expressed as functions of two vector variables P and Q (names chosen arbitrarily, since the functions are independent of the particular name choice for the origin and target spaces of the transform), while the latter four results are expressions using the variables given in L1 and L2.

*/

```

/*Catalog vector-space information given in
L1 and L2. Declare the arbitrary vector variables used in the
functional definitions to be P and Q, with same dimension as those
given in L1 and L2 */
SPACE(L1,SOURCESPACE),
SPACE(L2,DESTSPACE), DOMAIN:RANGE:[],
DUMMYSPACE:[P,''L1[2],Q,''L1[4]],
SPACE(DUMMYSPACE,DUMMY),
FOR I STEP 2 THRU LENGTH(L1) DO
  (DOMAIN:ENDCONS(L1[I],DOMAIN), RANGE:ENDCONS(L2[I],RANGE)),
  PRINT(DOMAIN,RANGE), OMU[0](P,Q):=AT(H(P,Q),[EP=0]),
  QONE:DUMMYDOMAIN[2][1],
  PU:RANGE[1], QU:RANGE[2],
/*Set up simplification and evaluation rules for av,ii,pb operators*/
LISTOFSIMPRULES:
  '[PB(FTRUE,GTRUE), PBSIMP(PB(FTRUE,GTRUE))],
  '[AV(FTRUE), AVSIMP(AV(FTRUE))],
  '[II(FTRUE), AVSIMP(II(FTRUE))],
  '[PB(FTRUE,GTRUE), PBVECSIMP(PB(FTRUE,GTRUE))]],
LISTOFEVALRULES:[['PBEVALUATION,PB(FTRUE,GTRUE),
  APPLY(PBEVAL,APPEND([FTRUE,GTRUE],DUMMYDOMAIN))],
  '[AVEVALUATION,AV(FTRUE),AEVAL(FTRUE)],
  '[IEVALUATION,II(FTRUE),IEVAL(FTRUE)]],
MATCHDECLARE([FTRUE,GTRUE,MTRUE],TRUE),
ADDONTORULES(LISTOFEVALRULES),
DYNAMALLOC:TRUE,KILL(LABELS), /*Turn on automatic storage allocation, free avstorage space devoted to previously
used labels (relabel expressions starting from zero)*/

```

```

/*For each order specified, a)isolate i-th order of Hamiltonian,
 b) compute and output S, K and coordinate transforms
 */
FOR I FROM 1 THRU N DO
  (OMU[I](P,Q):=COEFF(H(P,Q),EP^I),
   PRINT("ORDER ",I," TERMS ARE--"),PRINT(""),ADDONLISTOSIMPS(LISTOFSIMPRULES),
   PRINT("S IS ",SU[I]),SKIP(2),KNOCKOFFLISTOSIMPS(LISTOFSIMPRULES),
   OUTDISP('ESU[I],ESU[I],DUMMYDOMAIN),ADDONLISTOSIMPS(LISTOFSIMPRULES),
   STORE(SU),
   PRINT("K IS ",KU[I]),KNOCKOFFLISTOSIMPS(LISTOFSIMPRULES),SKIP(2),
   OUTDISP('EKU[I],EKU[I],DUMMYDOMAIN),
   STORE(KU),STORE(EKU),STORE(OMU),
   OUTDISP(SUPEROUT(DOMAIN[2],I),QLS[I],RANGE),
   OUTDISP(SUPEROUT(RANGE[2],I),QUS[I],DOMAIN),STORE(QUS),STORE(QLS),
   OUTDISP(SUPEROUT(DOMAIN[1],I),PLS[I],RANGE),
   OUTDISP(SUPEROUT(RANGE[1],I),PUS[I],DOMAIN),STORE(PLS),STORE(PUS)
  )
, DYNAMALLOC:FALSE)$

```

(C19) /*Supplemental functions for output, evaluation, and rule addition-deletion*/

SKIP(I):=FOR J THRU I DO PRINT("")\$

(C20) DUMMYDOMAIN:[P,Q]\$

(C21) ADDONTORULES(L):=BLOCK([I],FOR I:1 THRU LENGTH(L) DO APPLY(DEFRULE,L[I]))\$

(C22) REMOVEFROMRULES(L):=BLOCK([I],FOR I:1 THRU LENGTH(L) DO APPLY(REMRULE,L[I][1]))\$

(C23) OUTDISP(NAME, FUNC, DOM):=BLOCK([],PRINT(NAME,"(",DOM[1],",",DOM[2],""):="),
 SKIP(1),LDISP(APPLY(FUNC,DOM)),SKIP(3))\$

(C24) SUPEROUT(NAME,I):=IF I=0 THEN NAME^"0" ELSE
 IF I=1 THEN NAME^"1"
 ELSE NAME^"I"\$

```

(C26) /*FUNCTIONS FOR WHOLESALE ADDITION/DELETION OF SIMPLIFICATION, ETC. */
ADDSIMP(FUNCEXP,SIMP):=SIMPNAME[SIMP]:APPLY(TELLSIMPAFTER,[FUNCEXP,SIMP])$

(C27) ADDONLISTOFSIMPS(L):=FOR I THRU LENGTH(L) DO APPLY(ADDSIMP,L[I])$

(C28) REMOVESIMP(FUNCEXP,SIMP):=IF SIMPNAME[SIMP][1] # FALSE
    THEN SIMPNAME[SIMP]:APPLY(REMRULE,
        [INPART(FUNCEXP,0),
         SIMPNAME[SIMP][1]])
    ELSE ERROR("NO SUCH SIMP",SIMPNAME[SIMP][1])$

(C29) KNOCKOFFLISTOFSIMPS(L):=FOR I THRU LENGTH(L) DO APPLY(REMOVESIMP,L[I])$

(C31) /* Patterns and functions for componentwise pb in vectors */
PBVECSIMP(E):=IF PBPATO(E) # FALSE THEN MAP (COMPWISEPB,X) ELSE E$

(C32) COMPWISEPB(Y):=PBSIMP(PB(Y,FTRUE))$

(C33) MATCHDECLARE(LISTVAR,LISTP)$

MATCOM FASL DSK MACSYM being loaded
loading done

(C34) DEFMATCH(PBPATO,PB(LISTVAR,FTRUE))$

(C35) /* procedure to return a parameter list, given one argument (vector)
of the function domain of which the vector is a part of, i.e.
vparamlist(q) may return [p,q] if that has been declared through space
function (see below). */
VPARAMLIST(X):=BLOCK([G,I,ITEMS,SP,TYX],TYX:GET(X,TYPE),
    IF TYX = VECTOR
        THEN (SP:GET(X,SPACENAME),ITEMS:GET(SP,SPACEDESC),G:[],
            FOR I STEP 2 THRU LENGTH(ITEMS) DO
                G:ENDCONS(ITEMS[I],G),RETURN(G)))$

(C36) /*The use of the SPACE function declarres the dimensionality of certain
variables as vectors, and associates vectors with other vectors to be
considered as a space. This is necessary for proper evaluation of functions in situations in
which one parameter of the function is all that is given in customary notation
(such as dS/dQ, where S is a function of two or more arguments, both of which
are vectors.
The call is of the form SPACE(L,NAME),
Example:
SPACE([bigp,3,bigq,3],bigspace)

declares the bigspace is six dimensional, the first three components
collectively referred to as p, the second three as q. Thus, a particular
point in the space would be generally referred to as (p[1],p[2],p[3],q[1],q[2],q[3]).
```

This space declaration is necessary to insure proper function evaluation
when the arguments of varius functions are vectors
(like h(p,q):=p[1]+q[2]*p[3]^2).

```
/*
SPACE(L,NAME):=BLOCK([I],PUT(NAME,L,SPACEDESC),
    FOR I STEP 2 THRU LENGTH(L) DO
        (PUT(L[I],NAME,SPACENAME),PUT(L[I],VECTOR,TYPE),
         PUT(L[I],L[I+1],DIM)))$
```

(C37) /*VDIFF performs componentwise differentiation via vectors. F is a function (with vector arguments), x is the name of a vector which is an argument of f, e.g. VDIFF(F,X) is a vector

[dF/dX[1],dF/dX[2],...]

*/

```
VDIFF(F,X):=BLOCK([G,TYX,ANS,I],TYX:GET(X,TYPE),
/*if x is a vector, return a vector of derivatives, df/dx[i] */
IF TYX=VECTOR THEN
  (G:APPLY(F,VPARAMLIST(X)),ANS:[],FOR I:1 THRU GET(X,DIM) DO
    ANS:ENDCONS(DIFF(G,X[I]),ANS),RETURN(ANS))$
```

(C39) /* General "complexity" measure, attempting to drive expressions to simpler forms. A >> B if the expression A is more complicated than the expression B. The function "?great" is built in. */

REMOVE(">>",OPERATOR)\$

SYNEX FASL DSK MAXOUT being loaded
loading done

```
(C40) ">>"(A,B):=
BLOCK(
IF A=B THEN RETURN(FALSE),
IF ATOM(A) AND ATOM(B) THEN RETURN (?GREAT(A,B)),
IF ATOM(A) THEN RETURN(FALSE),
IF ATOM(B) THEN RETURN(TRUE),
RETURN(?GREAT(A,B)))$
```

(C41) /* make ">>" an infix operator, same binding as ">" */

INFIX (">>",80,80,ANY,ANY,CLAUSE)\$

```

(C42) /* set up simplification for pb "poisson bracket" */
/* first introduce predicate so that bilinearity of pb can be handled */
IPRED(X):=IS(NOT(ATOM(X)) AND INPART(X,0) = "+" AND (FTRUE:INPART(X,1),GTRUE:X-FTRUE,TRUE))$

(C43) /*Four simplification rules of PB implemented:
      a) bilinearity of PB (e.g. PB(a*f+b*g,c*h+z) =
          acPB(f,h) + bcPB(g,h) +
          aPB(f,z) + aPB(g,z), where a,b,c are
          constants, and f,g,h,z are functions)
      b) PB(f,g) = -PB(g,f) if f>>g (i.e. f more complex than g as ordered
          by complexity measure >> )
      c) Jacobi's rule PB(f,PB(g,h)) = -PB(h,PB(f,g))-PB(g,PB(h,f))
          if g>>f
      d) PB(f,f)=0

*/
MATCHDECLARE([FTRUE,GTRUE,HTRUE],TRUE)$

(C44) MATCHDECLARE(IPAT,IPRED)$

(C45) DEFMATCH(PBPAT1,PB(FTRUE,GTRUE))$

(C46) DEFMATCH(PBPAT2,PB(FTRUE,PB(GTRUE,HTRUE)))$

(C47) DEFMATCH(PBPAT3,PB(IPAT,HTRUE))$

(C48) DEFMATCH(PBPAT4,PB(HTRUE,IPAT))$

(C49) PBSIMP(E):=BLOCK([FTRUE,GTRUE,HTRUE,M,I,IPAT,PART1],
M:PBPAT1(E),
IF M = FALSE THEN RETURN(E),
IF (I:NUMFACTOR(FTRUE))#1 THEN IF I#0 THEN RETURN(I*PBSIMP(PB(FTRUE/I,GTRUE)))
ELSE RETURN(0),
IF (I:NUMFACTOR(GTRUE))#1 THEN IF I#0 THEN RETURN(I*PBSIMP(PB(FTRUE,GTRUE/I)))
ELSE RETURN(0),
IF FTRUE >> GTRUE THEN RETURN (-PBSIMP(PB(GTRUE,FTRUE)))
/* pb(f,g) ==> - pb(g,f) */
ELSE IF GTRUE=FTRUE THEN RETURN(0),
/* pb(f,f) ==> 0 */
IF PBPAT2(E)# FALSE AND GTRUE >> FTRUE THEN
RETURN(-PBSIMP(PB(HTRUE,PBSIMP(PB(FTRUE,GTRUE))))-PBSIMP(PB(GTRUE,PBSIMP(PB(HTRUE,FTRUE))))-
/* pb(f,pb(g,h)) ==> -pb(h,pb(f,g))-pb(g,pb(h,f)) */
ELSE
/* pb(a+b,c) ==> pb(a,c)+pb(b,c) */
IF PBPAT3(E)#FALSE THEN RETURN (PBSIMP(PB(FTRUE,HTRUE))+PBSIMP(PB(GTRUE,HTRUE)))
ELSE
/* pb(a,b+c) ==> pb(a,b)+pb(a,c) */
IF PBPAT4(E)#FALSE THEN RETURN(PBSIMP(PB(HTRUE,FTRUE))+PBSIMP(PB(HTRUE,GTRUE)))
ELSE /* return unchanged */ RETURN (E )$

(C50) /* Simplification rules for av and ii operators:
      a) linearity of AV and II
      b) av(ii(f))=0
      c) av(f*ii(g)) = -av(g*ii(f)) if g >> f
      d) av(pb(av(f),,g)) = pb(av(a),av(b))
      e) av(av(f)) = av(f)
      g) ii(pb(av(f),g)) = pb(av(f),ii(g)))
*/

```

```

(C51) DECLARE(II,LINEAR)$
(C52) DEFMATCH(AVPAT1,AV(FTRUE#II(GTRUE)))$
(C53) DEFMATCH(AVPAT2,AV(II(FTRUE)))$
(C54) DEFMATCH(AVPAT3,AV(PB(AV(FTRUE),GTRUE)))$
(C55) DEFMATCH(AVPAT3A,AV(PB(FTRUE,AV(GTRUE))))$ 
(C56) DEFMATCH(AVPAT4,AV(AV(FTRUE)))$ 
(C57) DEFMATCH(IIPAT1,II(PB(AV(FTRUE),GTRUE)))$ 
(C58) DEFMATCH(IIPAT1A,II(PB(FTRUE,AV(GTRUE))))$ 
(C59) AVSIMP(E):=BLOCK([FTRUE,GTRUE,II,
/* av(ii(x))=0 */
IF AVPAT2(E)#FALSE THEN RETURN(0)
ELSE /*av(a*ii(b)) ==> -av(b*ii(a)) if b>>a */
    IF AVPAT1(E)#FALSE THEN IF GTRUE>>FTRUE THEN RETURN (-AVSIMP(AV(GTRUE#II(FTRUE))))
        ELSE /* return unchanged */ RETURN (E)
    ELSE /* av(a) is independent of q1, so av(pb(av(a)),b) ==> pb(av(a),av(b)) */
        IF AVPAT3(E)#FALSE THEN RETURN ( PB(AV(FTRUE),AVSIMP(AV(GTRUE))))
        ELSE IF AVPAT3A(E)#FALSE THEN RETURN(PBSIMP(PB(AVSIMP(AV(FTRUE)),AV(GTRUE))))
            ELSE /* av(av(a)) = av(a) */
                IF AVPAT4(E)#FALSE THEN RETURN (AV(FTRUE))
                ELSE /* ii(pb(av(a),b)) ==> pb(av(a),ii(b)) */
                    IF IIPAT1(E)#FALSE THEN RETURN (PB(AV(FTRUE),AVSIMP(II(GTRUE))))
                    ELSE IF IIPAT1A(E) # FALSE THEN
                        RETURN(PBSIMP(PB(AVSIMP(AV(FTRUE)),AV(GTRUE))))
                    ELSE /* return unchanged */ RETURN(E)$

```

```

(C61) /*Definition of the evaluation rules for II, AV, and PB. AVEVAL, and IIEVAL
evaluate a function at a point, expand expressions until they become
sums of terms free of variable q[1] and/or terms with only a single
sin or cosine in them, then perform the AV or II operation on each term in the
sum. PBEVAL uses VDIFT and CRUSHDOT to perform vector differentiation and
dot products while not overstraining space requirements
for the dot product expressions. For all three procedures,
the routines TRIGMERGE, TRIGCRUNCH, and COLLECTTERMS are used to enforce
space economy.
*/

```

```

PBEVAL(F,G,P,Q):=BLOCK([A,B,I,C,LEN],
  IF VECTORFUNC(F)
    THEN (A:VDIFF(F,Q),B:VDIFF(G,P),C:[],LEN:LENGTH(A),
      FOR I THRU LEN DO C:ENDCONS(CRUSHDOT(A[I],B),C),A:VDIFF(F,P),
      B:-VDIFF(G,Q),
      FOR I THRU LEN DO C[I]:TRIGMERGE(C[I],CRUSHDOT(A[I],B)),
      RETURN(MAP(EV,APPLY(LAMBDA,['DUMMYDOMAIN,C]))))
    ELSE RETURN(MAP(EV,
      APPLY(LAMBDA,['DUMMYDOMAIN,
        TRIGMERGE(CRUSHDOT(VDIFT(F,Q),VDIFT(G,P)),
        CRUSHDOT(VDIFT(F,P),-VDIFT(G,Q))
        )]
      )
    )))

```

```

ANSLOT:0,LEN:LENGTH(A),
FOR IND:1 THRU LEN DO
    (ANSLOT:TRIGMERGE(ANSLOT,TRIGCRUNCH(A[IND]*B[IND]))),
RETURN(ANSLOT)$

(C63) AVEVAL(EXP):=BLOCK([I,A,NONX],
    IF FREEOF(QONE,EXP) THEN RETURN(EXP)
    ELSE (IF QONEINTRIG(EXP,QONE) THEN ( IF ODDP(MTRUE) THEN RETURN(0) ELSE RETURN(MULTTHRU(NONX,AVEVAL(ISOLATE(EXPAND(TRIGREDUCE(ANY)),QONE)))) )
    ELSE (A:INPART(EXP,0),
        IF A = "+"
        THEN RETURN(MULTTHRU(NONX*SUM(AVEVAL(INPART(EXP,I)/NONX),I,1,
        LENGTH(EXP))))
        ELSE (IF A = "*"
        THEN RETURN(MULTTHRU(NONX,AVPROD(ISOLATE(ANY,QONE))))
        ELSE RETURN(ERROR("AVEVAL CAN'T WORK ON",
        EXP))))))$

(C64) DECLARE(AVEVAL,LINEAR)$

(C65) AVPROD(EXP):=BLOCK([A],
    A:ISOLATE(MULTTHRU(EXP),QONE),
    IF NOT(PRODUCTP(A)) THEN RETURN(AVEVAL(A))
    ELSE(A:ISOLATE(EXPAND(TRIGREDUCE(A)),QONE),
        IF A#EXP THEN RETURN(AVEVAL(A))
        ELSE RETURN(ERROR("AVPROD CAN'T WORK ON",A))))$

(C66) DECLARE(AVPROD,LINEAR)$

(C67) PRODUCTP(X):=IS(NOT(ATOM(X)) AND INPART(X,0)="*")$

(C68) ODDP(X):=IS(INTEGERTP(X) AND NOT(INTEGERTP(X/2)))$

(C69) AEVAL(F):=MAP(EV,APPLY(LAMBDA,['DUMMYDOMAIN,TRIGCRUNCH(AVEVAL(APPLY(F,DUMMYDOMAIN))))])$

(C70) IIEVAL(F):=MAP(EV,APPLY(LAMBDA,['DUMMYDOMAIN,TRIGCRUNCH(IIEVAL(APPLY(F,DUMMYDOMAIN))))])$

(C71) IIEVAL(EXP):=BLOCK([A,NONX],
    IF FREEOF(QONE,EXP) THEN RETURN(0)
    ELSE (IF QONEINTRIG(EXP,QONE)
        THEN (IF MTRUE = 1
            THEN (IF TRIGFUNC=COS
                THEN RETURN(NONX*SIN(LININQONE)/CCOEF)
                ELSE RETURN(-1*NONX*COS(LININQONE)/CCOEF))
            ELSE RETURN(MULTTHRU(NONX,IIEVAL(ISOLATE(
            TRIGREDUCE(ISEEASINORCOS^MTRUE),QONE)))))

        ELSE (A:INPART(EXP,0),
            IF A = "+"
            THEN RETURN(MULTTHRU(NONX,SUM(IIEVAL(INPART(EXP,I)/NONX),I,1,
            LENGTH(EXP))))
            ELSE (IF A = "*"
            THEN RETURN(MULTTHRU(NONX,IIPROD(ANY)))
            ELSE RETURN(ERROR("IIEVAL CAN'T WORK ON",
            EXP))))))$

(C72) DECLARE(IIEVAL,LINEAR)$

(C73) IIPROD(EXP):=BLOCK([A],
    A:ISOLATE(MULTTHRU(EXP),QONE),
    IF NOT(PRODUCTP(A)) THEN RETURN(IIEVAL(A))
    ELSE(A:ISOLATE(EXPAND(TRIGREDUCE(ANY)),QONE),
        IF A#EXP THEN RETURN(IIEVAL(A))
        ELSE RETURN(ERROR("IIPROD CAN'T WORK ON",A)))

```

```

(C74) DECLARE(IIPROD,LINEAR)$

(C75) VECTORFUNC(F):=IF ATOM(F)
      THEN (IF FUNCTIONP(F)
            THEN (IF LISTP(APPLY(F,DUMMYDOMAIN)) THEN TRUE))
      ELSE (IF INPART(F,0) = LAMBDA
            THEN (IF LISTP(INPART(F,2)) THEN TRUE))$)

(C76) /*Various patterns are defined to help isolate trigonometric functions
occurring in expressions, necessary for av and ii evaluation as explained above,
also ffor TRIGCRUNCH, and TRIGMERGE (see below) */

NONZEROANDFREEOF(X,E):=IF E # 0 AND FREEOF(X,E) THEN TRUE$

(C77) TMP(X):=IS(X = SIN OR X = COS)$

(C78) TMP4(X,E):=IS(TRIGPAT4(E,X) # FALSE)$

(C79) TMP5(X,E):=IS(TRIGPAT5(E,X) # FALSE)$

(C80) LINP(X,E):=IS(LIN(E,X) # FALSE)$

(C81) QONEINTRIG(EXP,VAR):=IS(TRIGPAT3(EXP,VAR) # FALSE)$

(C82) MATCHDECLARE(TRIGFUNC,TMP,MTRUE,TRUE,LININQONE,LINP(X))$

(C83) MATCHDECLARE(CCOEF,NONZEROANDFREEOF(X),ECONST,FREEOF(X),ISEEASINORCOS,TMP5(X))$

(C84) MATCHDECLARE(SINORCOSTOAPOWER,TMP4(X))$

(C85) MATCHDECLARE(NONX,FREEOF(X),ANY,TRUE)$

(C86) DEFMATCH(RJFPAT3,ANY*NONX,X)$
NONX ANY PARTITIONS PRODUCT

(C87) QONEINTRIG(E,X):=IS(RJFPAT3(E,X)#FALSE AND TRIGPAT4(ANY,X)#FALSE)$

(C88) DEFMATCH(TRIGPAT4,ISEEASINORCOS^MTRUE,X)$

(C89) DEFMATCH(TRIGPAT5,TRIGFUNC(LININQONE),X)$

(C90) DEFMATCH(LIN,CCOEF*X+ECONST,X)$

(C92) /*EVALLIEB2 is the top level operator evaluator. It handles expressions
consisting of sums and products of functions and Operators, and applies the programmed
operator evaluation rules, then performs the specified arithmetic operations
(sums, products, and constant multiples) on the
functional results, and returns a single function (in parameters P qnd Q)
as a result. */

EVALLIEB2(X):=FUNCEXPEVALUNEVAL(
    APPLY(APPLYB2,['X,PBEVALUATION,AVEVALUATION,IIEVALUATION]),
    DUMMYDOMAIN)$

(C93) FUNCTIONP(X):=IS(MEMBER(FUNCTION,APPLY(PROPERTIES,[X])))$
```

```

(C94) /*Procedures for evaluation of functional expressions. Functions must
be all evaluated at a single point, the resulting algebraic expressions
combined into one expression, and that expression returned as a result*/
FUNCXPEVAL(X,DOM):=BLOCK([A,E,I],
  IF LENGTH(X) = 0
    THEN (IF CONSTANTP(X) THEN RETURN(X)
          ELSE (IF ATOM(X)
                  THEN (IF FUNCTIONP(X)
                          THEN RETURN(APPLY(X,DOM))
                          ELSE RETURN(X))))
    ELSE (A:INPART(X,0),
          IF A = "+"
            THEN RETURN(SUM(FUNCXPEVAL(INPART(X,I),DOM),I,1,
                            LENGTH(X)))
          ELSE (IF A = "***"
                  THEN RETURN(PROD(FUNCXPEVAL(INPART(X,I),
                                                DOM),I,1,
                                                LENGTH(X)))
                  ELSE (IF A = "##"
                          THEN (A:1,
                                FOR I FROM LENGTH(X) STEP
                                  -1 THRU 1 DO
                                    (E:INPART(X,I),
                                     A:FUNCXPEVAL(E,DOM)
                                     ^A),RETURN(A))
                          ELSE (IF A = LAMBDA
                                  THEN IF INPART(X,1)=DOM THEN RETURN(INPART(X,2)) ELSE
                                      RETURN(
                                        APPLY(X,DOM))
                                      ELSE RETURN(X))))))$)

(C95) /*FUNCXPEVAL takes an expression of functions, and returns a single
algebraic expression. FUNCXPEVALUNEVAL then LAMBDA-binds that
expression, returning a function as the result.*/
FUNCXPEVALUNEVAL(X,DOM):=IF NOT(ATOM(X)) AND INPART(X,0)=LAMBDA AND INPART(X,1)=DOM
  THEN X
  ELSE MAP(EV,APPLY(LAMBDA,['DOM',
                           TRIGCRUNCH(FUNCXPEVAL(X,DOM))]))$

(C96) APPLYUP(F,DOM):= IF NOT(ATOM(F)) AND INPART(F,0)= LAMBDA AND INPART(F,1)=DOM
  THEN INPART(F,2)
  ELSE APPLY(F,DOM)$

```

```

batch(crunch,?>)$

(C117) /*Functions and routines for simplifying expressions which are sums of
terms of the form

    a*sin(w), a*cos(w) or a,
where a may be a product or sum itself, free of sin or cosine terms,
w any arbitrary expressions. All the terms with similar sin or cos terms are
collected into one term, e.g. a sum

    a*sin(x)+b*sin(x)+a*cos(y)+d+e*cos(y)+f  is simplified
to      (a+b)*sin(x)+(a+e)*cos(y)+d+f

TRIGCRUNCH and TRIGMERGE attempt to expand products and sums
not of the proper form before using COLLECTTERMS to do the simplification,
using ordinary expansion and the multiple-angle formulas contained in the
built-in function TRIGREDUCE. The expansion is carried out cautiously so
that space requirements are reduced for intermediate
expressions.
*/
NONZEROANDFREEOFTRIG(X):=IF X#0 AND (FREEOF(SIN,X)) AND (FREEOF(COS,X)) THEN TRUE ELSE FALSE$

(C118) HASTRIG(X):=IS(NOT(FREEOF(SIN,X)) OR NOT(FREEOF(COS,X)))$

(C119) SUMP(X):=IS(NOT(ATOM(X)) AND INPART(X,0)="+"$)

(C120) MATCHDECLARE(NONTRIG,NONZEROANDFREEOFTRIG,TRIGGER,HASTRIG)$

(C121) DEFMATCH(TRIGPAT,NONTRIG*TRIGGER)$
TRIGGER NONTRIG PARTITIONS PRODUCT

(C122) TRIGCRUNCH(EXP):=BLOCK([FTRUE,NONTRIG,TRIGGER,TRIGFUNC,FACTORFLAG,ANS,A,B,I,J],
    ANS:0,FACTORFLAG:FALSE,
    IF SUMP(EXP)
        THEN IF MEMBER(FALSE,MAPLIST(OKAYTERM,EXP))
            THEN (A:0,
                FOR J:1 THRU LENGTH(EXP) DO
                    (A:TRIGMERGE(A,TRIGCRUNCH(INPART(EXP,J)))),
                    RETURN(A)
                )
            ELSE RETURN(COLLECTTERMS(EXP))
        ELSE IF PRODUCTP(EXP)
            THEN IF TRIGPATP(EXP)
                THEN (A:MULTTHRU(NONTRIG,
                    IF SUMP(TRIGGER) THEN TRIGCRUNCH(TRIGGER)
                    ELSE IF PRODUCTP(TRIGGER)
                        THEN IF MEMBER(FALSE,MAPLIST(TRIGTOAPP,TRIGGER))
                            THEN TRIGCRUNCH(MULTTHRU(TRIGGER))
                            ELSE EXPAND(TRIGREDUCE(EXPAND(TRIGGER)))
                        ELSE EXPAND(TRIGREDUCE(EXPAND(TRIGGER)))
                    ),
                    RETURN(COLLECTTERMS(A))
                )
            ELSE RETURN(EXP)
        ELSE RETURN(COLLECTTERMS(EXPAND(TRIGREDUCE(EXP))))
    )$
```

```

IF NOT(SUMP(A)) THEN RETURN(A)
ELSE WHILE(SUMP(A)) DO
  (I:INPART(A,1),
   IF TRIGPATP(I)
     THEN (ET:EXPAND(TRIGGER),
           B:COEFF(A,ET),A:SUBST(0,ET,A),
           IF (C:COEFF(A,TRIGGER))#0 THEN A:SUBST(0,TRIGGER,A),
           ANS:ANS+FACTOR(B+C)*ET)
     ELSE (ANS:ANS+FACTOR(I),A:EV(A-I))),
  IF TRIGPATP(A)
    THEN RETURN(ANS+FACTOR(NONTRIG)*EXPAND(TRIGGER))
  ELSE RETURN(ANS+FACTOR(A)))$

(C124) MATCHDECLARE([MTRUE,FTRUE],TRUE)$

(C125) TMP2(X):=IS (ANYTRIG(X)#FALSE)$

(C126) DEFMATCH(ANYTRIG,TRIGFUNC(FTRUE))$

(C127) TRIGPATP(X):=IS(TRIGPAT(X)#FALSE)$

(C128) TRIGMERGE(X,Y):=BLOCK([MERGED,C1,A,ET,NONTRIG,TRIGGER,TRIGFUNC,FTRUE],
MERGED:0,
WHILE SUMP(X) DO
  (A:INPART(X,1),
  IF TRIGPATP(A)
    THEN IF TMP2(TRIGGER)
      THEN (ET:EXPAND(TRIGGER),
            C1:COEFF(X,ET)+COEFF(Y,ET),
            MERGED:MERGED+FACTOR(C1)*ET,
            X:SUBST(0,ET,X),Y:SUBST(0,ET,Y)
            )
      ELSE ERROR(A," NOT IN PROPER FORM FOR TRIGMERGE",
                 X,Y)
    ELSE (MERGED:MERGED+FACTOR(A),X:EV(X-A))
  ),
RETURN(MERGED+COLLECTTERMS(X+Y))
)$

(C129) OKAYTERM(TERM):=IS(
  NOT HASTRIG(TERM) OR (TRIGPATP(TERM) AND ANYTRIG(TRIGGER)#FALSE))$

(C130) TRIGTOAPP(X):=IS(TRIGTOAP(X) # FALSE)$

(C131) DEFMATCH(TRIGTOAP,TRIGFUNC(FTRUE)^MTRUE)$

(C133)

```